Server-Side Parallel Data Reduction and Analysis

Daniel L. Wang, Charles S. Zender, and Stephen F. Jenks

University of California, Irvine, Irvine, CA 92697
{wangd,zender,sjenks}@uci.edu

Abstract. Geoscience analysis is currently limited by cumbersome access and manipulation of large datasets from remote sources. Due to their data-heavy and compute-light nature, these analysis workloads represent a class of applications unsuited to a computational grid optimized for compute-intensive applications. We present the Script Workflow Analysis for MultiProcessing (SWAMP) system, which relocates data-intensive workflows from scientists' workstations to the hosting datacenters in order to reduce data transfer and exploit locality. Our colocation of computation and data leverages the typically reductive characteristics of these workflows, allowing SWAMP to complete workflows in a fraction of the time and with much less data transfer. We describe SWAMP's implementation and interface, which is designed to leverage scientists' existing script-based workflows. Tests with a production geoscience workflow show drastic improvements not only in overall execution time, but in computation time as well. SWAMP's workflow analysis capability allows it to detect dependencies, optimize I/O, and dynamically parallelize execution. Benchmarks quantify the drastic reduction in transfer time, computation time, and end-to-end execution time.

1 Introduction

Despite the frenetic pace of technology advancement towards faster, better, and cheaper hardware, terascale data reduction and analysis remain elusive for most. Disk technology advances now enable scientists to store such data volumes locally, but long-haul network bandwidth considerations all but prohibit frequent terascale transfers. Bell et al. have noted that downloading data for computation is worthwhile only if the analysis involves more than 100,000 CPU cycles per byte of data, meaning that a 1GB dataset is only worth downloading if analysis requires 100 teracycles, or nearly 14 hours on a 2GHz CPU [1]. A typical case of evaluating global temperature change in 10 years requires averaging 8GB down to 330KB, and takes just 11 minutes to compute on a modern workstation, after spending over half an hour to download the input data over a speedy 30Mbits/s link. In data-intensive scientific analysis, data volume rather than CPU speed drives throughput, pointing to a need for a system that colocates computation with data.

Our Script Workflow Analysis for Multi-Processing (SWAMP) system provides a facility for colocating comput ation with data sources, leveraging shell

C. Cérin and K.-C. Li (Eds.): GPC 2007, LNCS 4459, pp. 744–750, 2007.
© Springer-Verlag Berlin Heidelberg 2007

script-based analysis methods to specify details through an interface piggy-backed over the Data Access Protocol (DAP) protocol [2]. Scripts of netCDF Operator (NCO) [3] commands are sent through an interface extended from DAP's subsetting facility and processed by a server-side execution engine. Resultant datasets may be retrieved in the same DAP request or deferred for later retrieval. The SWAMP execution engine additionally parses scripts for data-dependencies and exploits parallelism opportunities from the extracted workflow. By melding a computation service with a data hosting service, SWAMP eliminates data movement inefficiencies that are not addressed in current frameworks, which treat high data volume and high computational intensity as separate problems.

2 Background

The Grid computing field continues to grow rapidly in both hardware and software infrastructure. Computational grids offer highly parallel and distributed heterogeneous computing resources bound together by open standards, implemented by middleware such as the Globus Toolkit [4]. These grids are able to flexibly allocate resources and appropriately schedule generic applications, but are targeted towards large, compute-limited applications, such as grand challenges [5,6] where input data locality is not a primary scheduling concern. The Globus toolkit for grid systems allows users to define input and output files to be staged to and from compute nodes [7], but, as a generic system, does not detect when data movement costs exceed computational costs.

The Pegasus framework [8,9,10] leverages grid technology for complex data-dependent scientific workflows. Scientists use tools to specify workflows as directed acyclic task graphs containing data dependencies. Pegasus implements advanced resource allocation and locality-aware scheduling, but does not integrate with data services or apply automatic dependence extraction. Its locality-aware scheduling makes it worth considering for SWAMP backend processing.

Data grids focus on providing legible accessibility to terascale and petascale datasets with computational service limited to simple subsetting, if available. The Open-source Project for a Network Data Access Protocol (OPeNDAP) server serves a significant fraction of available geoscience data [2], and is the data service with which SWAMP integrates. The Earth System Grid II (ESG) project provides data via a later version of OPeNDAP (Hyrax), and is in the process of exploring the implementation of filtering servers that permit data to be processed and reduced closer to its point of residence [11]. We are exploring integration of SWAMP with ESG II data services. Other systems such as [12] [13] [14] exist to process or serve data in the geosciences data, but SWAMP differs from these projects in its shell-script interface and its focus on a class of application workflows that are data-intensive and compute-light.

3 Overview of SWAMP

The goal of the SWAMP system is to bring casual terascale computing to the average scientist. "Casual" implies that the system's interface must encourage

everyday usage, while "terascale" implies that the system's design must support terabyte data volumes. SWAMP is designed to support scientists' everyday shell scripts and supports high data volumes by shifting computation to data sources, trading expensive long-haul WAN bandwidth for relatively cheap LAN bandwidth. Computation efficiency is further enhanced by detecting and exploiting operator parallelization and I/O optimization opportunities in the scripted workflows. SWAMP differs from existing systems in its focus on a shell-script-based interface, aiming to derive data dependencies automatically with as little help from the scientist as possible. SWAMP also differs in its focus on data-intensive, compute-light workflows, targeting a class of data-heavy workflows where I/O, rather than CPU considerations dominate the decision to distribute computation.

3.1 Shell-Script Interface

The netCDF Operators (NCO) [15] are popular in the geoscience community for their ability to process gridded data at the granularity of files or sets of files, rather than single variables. This coarse granularity is crucial for practical analysis of the high volumes of data commonly resulting from satellite/surface measurements or Earth simulation runs. Because of their efficiency and ease at this scale, scientists commonly use compositions of these *operators* to describe their data analysis in shell scripts. SWAMP is unique in its ability to automatically parallelize shell-script execution through a custom parser that understands NCO command-line options and parameters. Special tags to flag intermediate (temporary) and output filenames are the only modifications needed. The resulting syntax, a subset of Bourne shell syntax, becomes a domain-specific language whose primitives are application binaries operating on files in a filesystem instead of variables in memory.

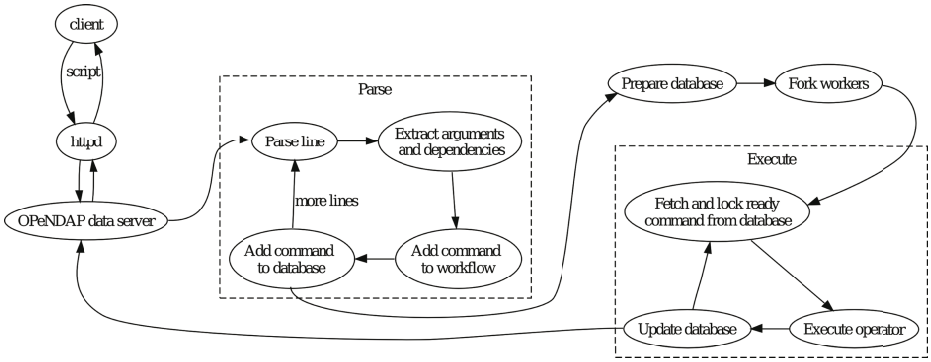


Fig. 1. SWAMP operation

3.2 Parallel Execution Engine

SWAMP scripts are processed on an execution engine implemented as an OPeNDAP data handler. This execution engine parses the user script for basic correctness and dataflow information, and manages execution of the script commands, optionally detecting and exploiting parallelism where available. Filenames are remapped to server-configured paths, and commands involving remote files are split into fetch and execute commands, allowing download to be overlapped with execution. Figure 1 summarizes parsing and execution in SWAMP.

Experience has shown that real scientific workflow scripts exhibit significant script-line-level parallelism. To exploit this, SWAMP builds a dependency tree at parse time. Initially, the only commands ready to execute are the tree roots, but as commands finish, dependent commands which have no unfinished parents become ready as well. After parsing, SWAMP forks off worker processes to begin parallel script execution. Workers cooperate in a peer model, communicating and preventing duplicate work by updating execution state in a shared relational database, currently SQLite. Thus, we can satisfy n-wide execution as long as n commands are ready to execute. SWAMP's SQLite database is stored in a standard Linux tmpfs RAM-based filesystem. Originally, the database was stored on standard disk, but performance suffered greatly due to I/O contention in concurrent execution modes.

4 Results

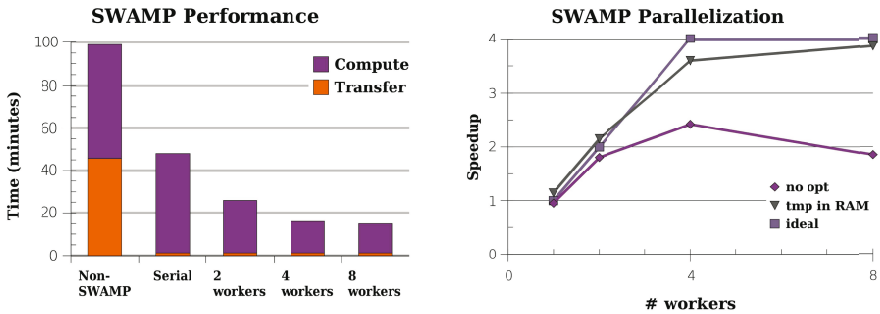
4.1 Test Setup

We tested our system with a script that resamples Community Atmospheric Model simulation data into time-steps that can be better compared against observed NASA Quick Scatterometer (QuikSCAT) data [16]. In this script, ten years of data at 20-minute timesteps are masked for their surface wind speed values at 6:00AM and 6:00PM, the local times from the QuikSCAT satellite passes. The script contains over 14,000 NCO command-lines for masking, averaging, concatenating, and editing, which produce 228MB of resultant data from 8230MB of input data, and generate 26GB of temporary intermediate files in the process.

We tested our system on a dual Opteron 270 with 16GB of memory with dual 500GB SATA drives in RAID 1, running CentOS 4.3 Linux. Figure 2 summarizes the test results. Transfer times listed are estimated assuming 3MBytes/s ($3*2^{20}$) bandwidth, based on NPAD *pathdiag* [17] measurement of 30Mbits/s bandwidth between UCI and the National Center for Atmospheric Research(NCAR). In our example, a scientist can avoid downloading nearly 8GB, obtaining just 228MB of output rather than the entire input dataset and saving 46 minutes of transfer time. Our baseline case shows the execution time of the original shell script and the time to download the input data, and takes 99 minutes overall.

4.2 Performance Gain

Test results are summarized in Figure 2. Figure 2(b) shows that SWAMP's overhead over baseline is slight, with parse and script analysis increasing computational time by 14% (1 worker case, no opt), but more than compensated when I/O optimization is enabled. Figure 2(a) shows the domination of transfer time savings, along with the parallezation benefit that is only through SWAMP's unique script dependency extraction. Parallelization easily saturates the test system's four CPU cores, bringing overall time from 99 minutes without SWAMP to 16 minutes with SWAMP configured for four workers, giving a 6x performance gain.



(a) Overall I/O-optimized performance (b) Parallelization speedup

Fig. 2. SWAMP performance

4.3 I/O Optimization

In Figure 2(b), we compare the performance of SWAMP with varying numbers of worker processes and toggling intermediate file optimization. Heavy I/O contention was obvious in early testing, leading to our development of a mechanism for explicitly storing intermediate files in a tmpfs (ramdisk-backed) filesystem rather than a disk-backed filesystem. Referring to Figure 2(b), we see that the performance degradation with a disk-backed filesystem at 8 workers is significant (≈24% relative to 4 workers), but eliminated by our I/O optimization. With this simple optimization, we see SWAMP's performance closely tracking an ideal speedup curve.

4.4 Summary

Our system targets scientists with compute capacity or network connectivity less than what a data center offers, which we believe should include most scientists. Data centers should benefit as well from reduced external network usage, which is often more costly than computational capacity. Our tests quantify the significant savings in bandwidth usage and the corresponding transfer time due to the relocation of computation off the desktop. Our tests also show the potential performance increase which is enabled by simple analysis of scripts.

5 Conclusion

A server-side data reduction and analysis system saves scientists time and bandwidth, enabling them to exploit potentially greater computing resources with minimal additional effort. We have leveraged existing script-based methods of analysis and the widely used DAP protocol to provide simple distributed computing to non-computer-scientists. Combining computation with data services has drastically reduced network transfer, and exploiting script-level parallelism has yielded linear speedup with CPU count, thus yielding a 6 times performance improvement in our test. Our tests have also shown the importance of I/O issues in data intensive workflows, quantifying the performance degradation and offering a possible solution. While performance of the current implementation already provides a significant speedup, future implementations will further exploit clustering and parallelism available at the data center, further enhancing performance. Systems such as ours that colocate computation with data will be well poised to meet the demands of more comprehensive, more detailed, and more frequent analyses, and will facilitate data-intensive science.

Acknowledgments

The authors would like to thank Scott Capps, whose research makes use of the above workflow. This research is supported by the National Science Foundation under Grants ATM-0231380 and IIS-0431203.

References

1. Bell, G., Gray, J., Szalay, A.: Petascale computational systems. IEEE Computer **39**(1) (2006) 110–112
2. Cornillon, P.: OPeNDAP: Accessing data in a distributed, heterogeneous environment. Data Science Journal **2** (2003) 164–174
3. Zender, C.S.: netCDF Operators (NCO) for analysis of self-describing gridded geoscience data. Submitted to Environ. Modell. Softw. (2006) Available from `http://dust.ess.uci.edu/ppr/ppr_Zen07.pdf`.
4. Foster, I., Kesselman, C.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, San Francisco, CA (1998)
5. Feigenbaum, E.A.: Some challenges and grand challenges for computational intelligence. J.ACM **50**(1) (2003) 32–40
6. Gray, J.: What next?: A dozen information-technology research goals. J.ACM **50**(1) (2003) 41–57
7. Foster, I., Kesselman, C.: Globus: A Metacomputing Infrastructure Toolkit. International Journal of Supercomputer Applications **11**(2) (1997) 115–128
8. Maechling, P., Chalupsky, H., Dougherty, M., Deelman, E., Gil, Y., Gullapalli, S., Gupta, V., Kesselman, C., Kim, J., Mehta, G., Mendenhall, B., Russ, T., Singh, G., Spraragen, M., Staples, G., Vahi, K.: Simplifying construction of complex workflows for non-expert users of the southern california earthquake center community modeling environment. SIGMOD Rec. **34**(3) (2005) 24–30

9. Singh, G., Deelman, E., Mehta, G., Vahi, K., Su, M.H., Berriman, G.B., Good, J., Jacob, J.C., Katz, D.S., Lazzarini, A., Blackburn, K., Koranda, S.: The pegasus portal: web based grid computing. In: SAC '05: Proceedings of the 2005 ACM symposium on Applied computing, New York, NY, USA, ACM Press (2005) 680–686
10. Deelman, E., Singh, G., Su, M.H., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Vahi, K., Berriman, G.B., Good, J., Laity, A., Jacob, J.C., Katz, D.S.: Pegasus: A framework for mapping complex scientific workflows onto distributed systems. Scientific Programming **13**(3) (2005) 219–238
11. Bernholdt, D., Bharathi, S., Brown, D., Chanchio, K., Chen, M., Chervenak, A., Cinquini, L., Drach, B., Foster, I., Fox, P., Garcia, J., Kesselman, C., Markel, R., Middleton, D., Nefedova, V., Pouchard, L., Shoshani, A., Sim, A., Strand, G., Williams, D.: The earth system grid: Supporting the next generation of climate modeling research. Proceedings of the IEEE **93**(3) (2005) 485–495
12. Abramson, D., Kommineni, J., McGregor, J.L., Katzfey, J.: An atmospheric sciences workflow and its implementation with web services. Future Gener. Comput. Syst. **21**(1) (2005) 69–78
13. Woolf, A., Haines, K., Liu, C.: A Web Service Model for Climate Data Access on the Grid. International Journal of High Performance Computing Applications **17**(3) (2003) 281–295
14. Chen, L., Agrawal, G.: Resource allocation in a middleware for streaming data. In: Proceedings of the 2nd workshop on Middleware for grid computing, New York, NY, USA, ACM Press (2004) 5–10
15. Zender, C.S.: NCO User's Guide, version 3.1.4. http://nco.sf.net/nco.pdf (2006)
16. Tsai, W.Y., Spencer, M., Wu, C., Winn, C., Kellogg, K.: SeaWinds on QuikSCAT: Sensor Description and Mission Overview. In: Proceedings of the IEEE International Geoscience and Remote Sensing Symposium. Volume 3., Honolulu, HI (2000) 1021–1023
17. Mathis, M., Heffner, J., Reddy, R.: Web100: extended tcp instrumentation for research, education and diagnosis. SIGCOMM Comput. Commun. Rev. **33**(3) (2003) 69–79